**Chapter one: introduction**

**This chapter will discuss the following concepts:**

**1-1 Introduction**

**1-2 Software engineering**

**1-3 Essential Attributes of Good Software**

**1-4 Software process**

**1-5 Computer Science and System Engineering**

**1-6 Software engineering diversity**

**1-7 Software engineering and the Web**

**1-1 introduction**

 The term software engineering is composed of two words, **software** and **engineering**.

 **Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be a collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.

 **Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods.

IEEE defines software engineering as:

 The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software. The experience is arranged in the form of methodologies and guidelines. A small program can be written without using software engineering principles. But if one wants to develop a large software product, then software engineering principles are absolutely necessary to achieve a good quality software cost effectively. Without using software engineering principles it would be difficult to develop large programs.

 In industry it is usually needed to develop large programs to accommodate multiple functions. A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes. Software engineering helps to reduce this programming complexity. Software engineering principles use two important techniques to reduce problem complexity: abstraction and decomposition.

 The principle of abstraction implies that a problem can be simplified by omitting irrelevant details. In other words, the main purpose of abstraction is to consider only those aspects of the problem that are relevant for certain purpose and suppress other aspects that are not relevant for the given purpose. Once the simpler problem is solved, then the omitted details can be taken into consideration to solve the next lower level abstraction, and so on. Abstraction is a powerful way of reducing the complexity of the problem. The other approach to tackle problem complexity is decomposition. In this technique, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one. However, in this technique any random decomposition of a problem into smaller parts will not help. The problem has to be decomposed such that each component of the decomposed problem can be solved independently and then the solution of the different components can be combined to get the full solution. A good decomposition of a problem should minimize interactions among various components. If the different subcomponents are interrelated, then the different components cannot be solved separately and the desired reduction in complexity will not be realized.

**1-2 software engineering**

 **Software engineering** is an engineering discipline that concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

 Software engineers are concerned with developing software products (i.e., soft-ware which can be sold to a customer). There are two kinds of software products:

1. **Generic products**: These are stand-alone systems that are produced by a development organization and sold on the open market to any customer who is able to buy them. Examples of this type of product include software for PCs such as databases, word processors, drawing packages, and project-management tools. It also includes so-called vertical applications designed for some specific purpose such as library information systems, accounting systems, or systems for maintaining dental records.

2. **Customized products**: These are systems that are commissioned by a particular customer. A software contractor develops the software especially for that customer. Examples of this type of software include control systems for electronic devices, systems written to support a particular business process, and air traffic control systems.

**1-3 Essential Attributes of Good Software**

1. ***Maintainability***: Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of changing business environment.

2. ***Dependability and security***: Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.

3. ***Efficiency***: Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.

4. ***Acceptability:*** Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

**1-4 Software process**

 The systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

1. **Software specification**: where customers and engineers define the software that is to be produced and the constraints on its operation.

2. **Software development**: where the software is designed and programmed.

3. **Software validation**: where the software is checked to ensure that it is what the customer requires.

4. **Software evolution**: where the software is modified to reflect changing customer and market requirements.

**1-5 Computer Science and System Engineering**

 Software engineering is related to both computer science and systems engineering:

1. Computer science is concerned with the theories and methods that underlie computers and software systems, whereas software engineering is concerned with the practical problems of producing software.
2. System engineering is concerned with all aspects of the development and evolution of complex systems where software plays a major role. System engineering is therefore concerned with hardware development, policy and process design and system deployment, as well as software engineering. System engineers are involved in specifying the system, defining its overall architecture, and then integrating the different parts to create the finished system.

**1-6 Software engineering diversity**

There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.

The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

There are many different types of application including:

1. Stand-alone applications: These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.

2. Interactive transaction-based applications: Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.

3. Embedded control systems: These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system.

4. Batch processing systems: These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.

5. Entertainment systems: These are systems that are primarily for personal use and which are intended to entertain the user.

6. Systems for modeling and simulation: These are systems that are developed by scientists and Engineers to model physical processes or situations, which include many, separate, interacting objects.

7. Data collection systems: These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.

8. Systems of systems: These are systems that are composed of a number of other software systems.

**1-7 Software engineering and the Web**

The development of the World Wide Web has had a profound effect on all of our lives. Initially, the Web was primarily a universally accessible information store and it had little effect on software systems. These systems ran on local computers and were only accessible from within an organization. Around 2000, the Web started to evolve and more and more functionality was added to browsers. This meant that web-based systems could be developed where, instead of a special-purpose user interface, these systems could be accessed using a web browser. the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational software. Instead of writing software and deploying it on users’ PCs, the software was deployed on a web server. This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC.

The next stage in the development of web-based systems was the notion of web services. Web services are software components that deliver specific, useful functionality and which are accessed over the Web. Applications are constructed by integrating these web services, which may be provided by different companies. In principle, this linking can be dynamic so that an application may use different web services each time that it is executed.

In the last few years, the notion of ‘software as a service’ has been developed. It has been proposed that software will not normally run on local computers but will run on ‘computing clouds’ that are accessed over the Internet. If you use a service such as web-based mail, you are using a cloud-based system. A computing cloud is a huge number of linked computer systems that is shared by many users. Users do not buy software but pay according to how much the software is used or are given free access in return for watching adverts that are displayed on their screen.

The advent of the web, therefore, has led to a significant change in the way that business software is organized. Before the web, business applications were mostly monolithic, single programs running on single computers or computer clusters. Communications were local, within an organization. Now, software is highly distributed, sometimes across the world. This radical change in software organization has, obviously, led to changes in the ways that web-based systems are engineered. For example:

1. Software reuse has become the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.

2. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems should be developed and delivered incrementally.

3. User interfaces are constrained by the capabilities of web browsers. Application interfaces on web-based systems are often poorer than the specially designed user interfaces on PC system products.